Forces on Architecture Decisions – A Viewpoint

Heesch, Uwe van; Avgeriou, Paris; Hilliard, Rich

Published in:
EPRINTS-BOOK-TITLE

Publication date:
2012

Forces on Architecture Decisions – A Viewpoint

Uwe van Heesch
University of Groningen,
Fontys University of Applied Sciences
Venlo, The Netherlands
uwe@vanheesch.net

Paris Avgeriou
University of Groningen
Groningen, The Netherlands
paris@cs.rug.nl

Rich Hilliard
Freelance software systems architect
USA
r.hilliard@computer.org

***Abstract*— In this paper, the notion of *forces* as influences upon architecture decisions is introduced. To facilitate the documentation of forces as a part of architecture descriptions, we specify a *decision forces viewpoint*, which extends our existing framework for architecture decisions, following the conventions of the international architecture description standard ISO/IEC/IEEE 42010. The applicability of the viewpoint was validated in three case studies, in which senior software engineering students used it to document decisions in software projects; two of which conducted for industrial customers. The results show that the forces viewpoint is a well-received documentation approach, satisfying stakeholder concerns related to traceability between decision forces and architecture decisions.**

I. Introduction

Decisions, and the rationale for those decisions, are pervasive elements of software architecture [1]. Because of their crucial role, architecture decisions and rationale need to be captured and managed throughout the lifetime of a software architecture, as with any other important part of the architecture documentation. Moreover, decisions and their rationale should be documented in a form that integrates with the documentation of other types of architecture information in order to provide traceability between decisions and those other types.

Kruchten proposed to capture the rationale behind an architecture using architecture decisions as first-class entities of architecture description [1]. To date, different approaches have been presented to practically realize the documentation of architecture decisions; prominent among those are decision templates, as introduced by Tyree and Akerman [2] (see [3] for a discussion of various decision documentation approaches).

ISO/IEC/IEEE 42010 [4] addresses the areas of recording architecture decisions and architecture rationale as part of an architecture description, specifying general requirements for decision documentation, but not particular mechanisms. As with any other kind of architecture information, architecture decisions and rationale pertain to different stakeholders' concerns. Consequently, a single form of representation is often not applicable to all concerns in a usable form; instead different forms of representation, arranged as *architecture views*, can each effectively address a subset of concerns.

Since the earliest work on the foundations of software architecture by Perry and Wolf [5], and exemplified by Kruchten's 4+1 model [6], the idea of documenting software architecture using multiple views has been widely adopted. IEEE Std 1471:2000 [7] first codified this practice of multiple views, with each view addressing specific concerns of interest to system stakeholders and introducing *viewpoints* to establish the conventions used in each view.

Building on this practice, in our previous work, we introduced a documentation framework for architecture decisions using the conventions of ISO/IEC/IEEE 42010, containing an initial set of four viewpoints for architecture decisions: a decision detail viewpoint, a decision relationship viewpoint, a decision chronology viewpoint and a decision stakeholder involvement viewpoint, each dedicated to specific decision-related concerns [3] (an example of a decision-related concern is *What decisions are dependent on decision D?*.)

In this paper, we extend our earlier framework with the decision forces viewpoint (or shortly *forces viewpoint*), which is dedicated to establishing traceability between architecture decisions, stakeholder concerns and the forces driving the decisions. Forces, in this context, include traditional requirements, but they also take the experience and expertise of the development team, as well as business and projects constraints, into account. A force, in short, is a broad concept, capturing anything that has a potential non-trivial impact of any kind on an architect when making decisions.

The forces viewpoint was validated in three case studies conducted with groups of senior students. Two of the groups worked independently on industrial software projects; the third group started an open source project as part of a module on Java EE. The results are promising, as they show that the forces viewpoint is well-received by the students, while satisfying many decision-related stakeholder concerns. Further, we learned that the forces viewpoint supports students in following a systematic and rational decision making process, when being created iteratively during the architecting process.

The rest of this paper is organized as follows. Section II introduces the viewpoint framework and the basic ideas behind ISO/IEC/IEEE 42010. In Section III, the decision forces viewpoint is specified. Section IV reports on the case studies conducted to validate the viewpoint. In the next section, we briefly outline related work. Finally, in Section VI, we conclude and present areas for future work.

II. A framework for architecture decisions

In this section, the main ideas behind ISO/IEC/IEEE 42010 [4] and the framework for architecture decisions [3], which

978-0-7695-4827-2/12 $26.00 © 2012 IEEE
DOI 10.1109/WICSA-ECSA.212.18

were the basis for the development of the decision forces viewpoint, will be briefly introduced.

A. ISO/IEC/IEEE 42010

ISO/IEC/IEEE 42010 is an international standard for the description of software architectures (and other kinds of system architectures). It is based on a few principles:

1) an *architecture description* (AD) expresses an architecture (of a system or other entity of interest);
2) an AD addresses the concerns of the system's stakeholders for that architecture.
3) the concerns drive the selection of the representation conventions (called *viewpoints*) used to express the architecture, each of which is dedicated to framing specific concerns;
4) consistency between the views is maintained using correspondence rules.

Building upon these principles, ISO/IEC/IEEE 42010 defines the required contents of individual ADs, the form of architecture description languages, and architecture frameworks.

B. Four viewpoints for architecture decisions

The framework for architecture decisions, introduced in our previous work [3], consists of an initial set of four viewpoints, each of which being dedicated to satisfying specific stakeholder concerns related to architecture decisions.

The **decision relationship viewpoint** makes relationships between architecture decisions explicit. Examples of decision relationships are *is caused by*, *depends on*, or *is alternative to*. Apart from relationships, views using this viewpoint document the current state of each decision in the system (e.g., *decided*, *approved*, or *rejected*). The **stakeholder involvement viewpoint** explains the responsibilities of specific stakeholders in the decision-making process. For example, views of this viewpoint show the stakeholders who proposed, confirmed, or validated particular decisions. The **decision chronological viewpoint** shows the evolution of architecture decisions over time. It also depicts architecture iterations and their endpoints (typically milestones, snapshots, or releases). The chronological viewpoint is the only viewpoint with a temporal component. All other types of views *freeze* a specific state of the architecture.

Whereas the previously mentioned viewpoints focus on specific aspects of architecture decisions to optimally frame their related concerns, the **decision detail viewpoint** is an aggregate viewpoint. This viewpoint combines the information shown in all other viewpoints, by giving detailed information about single architecture decisions. The detail viewpoint's model kind (a model kind establishes the conventions for all models in the respective view), at the same time, acts as a shared metamodel for all viewpoints in the framework.

The decision forces viewpoint, introduced in this paper, extends this existing set of viewpoints focusing on traceability between architecture decisions, stakeholder concerns, and decision forces. It integrates seamlessly into the decision framework and its shared metamodel.

III. Decision forces viewpoint

Views using the decision forces viewpoint make explicit the relationships between architectural decisions and the forces that influenced the architect when making the decisions out of multiple alternatives. The term *force* is taken from the pattern community, which uses forces to elaborate on the description of a problem to be solved by a pattern's proposed solution. They define a force as "[...] any aspect of the problem that should be considered when solving it." [8]. Similarly, when considering architecture decisions, a *force* is any aspect of an architectural problem arising in the system or its environment (operational, development, business, organizational, political, economic, legal, regulatory, ecological, social, etc.), to be considered when choosing among the available decision alternatives.

Forces arise from many sources; most often from requirements, but also from constraints, architecture principles and other "intentions" imposed upon the system; including personal preferences or experience of the architect(s) and the development team; and business goals such as quick-time-to-market, low price, or strategic orientations towards specific technologies (see [9] for an empirical study on influence factors on software architecture). Before making decisions, the architect assembles all forces relevant in the context of the system to be developed. It can be a good practice to maintain a list of typical domain-specific forces from different projects to make sure that not important forces are forgotten.

Different forces may be orthogonal to one another, they may support, antagonize or contradict each other. Therefore, an architect must balance forces to make the best possible decisions. Figure 1 shows an extract from a decision forces view, which was created as part of a pilot study conducted to validate the design of the case studies reported below. In the pilot study, the decision viewpoints from the previously mentioned framework [3] and the decision forces viewpoint were used to document architecture decisions made in a non-academic distributed open source online banking and accounting system for small and medium-sized companies.

The left part of the table shows the forces that were considered when choosing among the decision alternatives listed across the top of the table. Each force is classified by one or more concerns (please refer to Section III-B for an explanation of the relationship between forces and concerns). The decision alternatives can be grouped into decision topics (e.g. *view technology*, or *data storage* in Figure 1), if they were taken into consideration as alternatives to solve a particular problem. Within a decision topic, there can only be one decision with a state equal to or higher than *decided* (please refer to [3] for a description of all decision states). The comment box in Figure 1 contains an example of a textual description of a force-decision combination. The pluses and minuses indicate a positive or negative impact of a force on a decision alternative; an empty field means that a force is not applicable or neutral; a question mark expresses uncertainty (please refer to Section III-A for a more detailed description of

Decision Forces				View technology			Data storage	Middleware	DBMS	
				<decided>	<discarded>	<discarded>	<decided>	<discarded>	<discarded>	<decided>
				Java Swing	PHP	JSF	Central DS	EJB	MySQL	PostgreSQL
	Architecture significant requirements									
	Code	Description	Concern(s)							
	R1	Avg. response time <= 0.1s	Time behavior	++	+	-	-	-	+	+
	R5	Integrate mult. payment providers	Extendability	+		+		+		
	R6	Reliability of data storage	Reliability				++	+	+	++
	R8	Availability of full service (99.9%)	Reliability	++	+	+	+	-	+	+
	R9	Support growing no of users	Scalability	++	+	-	-	+	-	?
	R13	Security (personal data protection)	Security	+		?	+		?	?
	R16	Client platform independence	Portability	+	++	++				
	R23	Operability of user interface	Usability	++	+	+				
	R24	Communication via Internet	Network comm.		++	++	+	+	+	+
	R26	HBCI support	Banking protocols	+	?	+		+		
	R27	No licence costs	Development costs	+	+	+			++	++
	Other forces									
	F1	Inhouse experience	Development time							
	F1.1	Swing (very good)	Development time	++						
	F1.2	PHP (decent)	Development time		+					
	F1.3	JPA (good)	Development time							
	F1.4	MySQL (very good)	Development time				+		++	
	F1.5	JSF (very good)	Development time			+				
	F2	Strategic knowledge development	Competitiveness							
	F2.1	Learn Postgres	Competitiveness				+			++
	F2.2	Improve Javascript skills	Competitiveness	--	+	+				
	F2.3	Learn Jquery	Competitiveness	--	+	+				
	F4	Linux server available	Development costs		+	+	+	+		+
	F5	Non business criticality	Business criticality							+
	F7	Resource usage on server	Resource utilization	++	-	--	--	--	+	?

No resources needed. All calculations are performed on client side.

Figure 1. Excerpt from a decision forces view (see III-A for conventions used here)

the ratings). The architect evaluates each architectural decision alternative in the context of the forces. As a result of the evaluation, a force can have a positive, negative, currently unknown, or neutral impact on the architect with respect to a decision; it either attracts the decision maker towards a specific decision alternative, or it repels the decision maker from an alternative, or it has no effect. Figure 2 illustrates the application of forces on an architect when choosing between two database management systems. On the one hand, the development team has a lot of experience using MySQL; this force attracts the architect towards choosing MySQL. On the other hand, the company wants to develop strategic knowledge with PostgreSQL, which is also more reliable than MySQL and turns out to scale better. In this particular case, after balancing these forces, the architect would probably choose PostgreSQL, provided that no other decision alternatives were taken into consideration. In a more general case, an architect would need to decide between more than two options.

A. Forces Viewpoint Specification

Table I lists the decision-related concerns[1] framed by the decision forces viewpoint. These decision-related concerns are a subset of a larger set of concerns identified in our previous work [3]. The codes in Table I were copied from [3] for consistency.

Views of the decision forces viewpoint are dedicated to supporting decision–force traceability. They can be used by stakeholders interested in decision rationale, decisions relevant for specific stakeholder concerns, addressed requirements, conflicting forces and how these all relate to each other. The main stakeholders for this viewpoint are architects, but also reviewers and other stakeholders who need to comprehend the choices made in the architecture. Table II shows the stakeholders along with their main decision-related concerns with respect to the forces viewpoint. Similarly to the decision-

[1]The term *decision-related* concern is used to refer to concerns pertaining to decision documentation (as opposed to any other types of stakeholder concerns which are simply termed *concerns*).

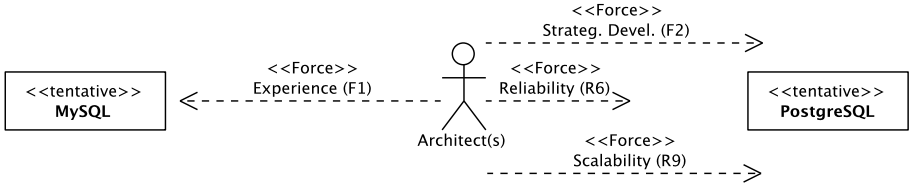

Figure 2. Application of forces on an architect

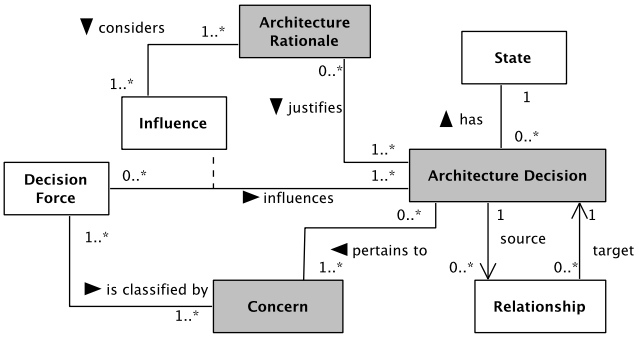

Figure 3. Metamodel of decision forces viewpoint

related concerns, the stakeholders were identified in our previous work.

Table I
CONCERNS OF THE DECISION FORCES VIEWPOINT

Code	Concern
C3	What is the rationale for decision D?
C4	What concerns C_i does decision D pertain to?
C5	What forces F_j impact/influence decision D?
C6	What decisions D_k are influenced by force F?
C7	What forces F_l have conflicting influences on decision D?
C23	What decisions D_p or decision sub-graphs SG_q can be reused in other projects?

Table II
TYPICAL STAKEHOLDERS OF THE DECISION FORCES VIEWPOINT AND THEIR CONCERNS

Stakeholder	Concerns
Architect	C3, C4, C5, C6, C7
Reviewer	C3, C4, C5, C6, C7
Requirements Engineer	C4, C6, C7
New project member	C3
Domain expert	C23

The decision forces viewpoint consists of a single model kind. Figure 3 depicts its metamodel, which presents the conceptual elements for architecture models that adhere to it. This model is part of a shared metamodel, which is used by all viewpoints of the decision documentation framework. Together with well-defined correspondence rules, the shared metamodel ensures consistency among the views of different viewpoints.

The elements in Figure 3 with a gray background map to the corresponding elements in Figures 2 and 4 of ISO/IEC/IEEE 42010. In the following, each of the elements used in Figure 3 is briefly described.

An architecture decision pertains to one or more concerns. Forces views show only the current state of each decision (e.g. *decided*, or *discarded* [3]). While decisions can generally have different types of relationships with each other, the forces viewpoint only regards the *is alternative for*-relationship to group multiple decision alternatives into a decision topic.

According to ISO/IEC/IEEE 42010, "Architecture rationale captures explanation, justification or reasoning about architecture decisions that have been made." [4]. In terms of the forces viewpoint, the architecture rationale should balance all relevant forces that influence a decision. Note that architecture rationale is not described in forces views; it is documented explicitly in decision detail views, which are part of the decision framework. In the forces viewpoint's model kind, the association between Architecture Rationale and Influence implies that the rationale description should consider the relevant forces.

All forces are classified by one or more concerns. A stakeholder could for instance be concerned about development cost, while concrete forces classified by this concern could be "not to use paid 3rd-party licenses", or to "use available hardware where possible". The force not to use 3rd-party licenses could, besides the development cost concern, be classified by a legal concern (e.g how the software can be distributed).

Apart from a textual qualification, the *influences* relationship between decision force and architecture decision can take one of the following values, estimated by the architect(s) of the system:

- ++: A force strongly supports a specific decision alternative to be chosen. An example from Figure 1 is the operability force, which strongly advocates the choice of Swing/Java, because Swing can be used to develop rich graphical user interfaces.
- +: A force moderately supports an alternative.
- blank: A force has a neutral influence on a decision alternative, or it is not applicable.
- -: A force moderately opposes an alternative.
- - -: A force strongly opposes an alternative to be chosen. For instance, if the programming team has no experience in functional programming, then this would be a strong argument against choosing Lisp or Haskell as a programming language.
- X: A decision alternative is prevented by a force. For instance, a force could be not to use libraries distributed under an open source license. Such a force would for instance prevent the use of Apache Lucene as a search library. Nevertheless, it can make sense to document such a decision alternative, because the forces view could be used to negotiate constraints or requirements with the customer, if its advantages clearly outweigh the opposing forces.
- ?: It is currently unclear how the decision alternative is impacted by a force. This rating should be temporary, indicating that prototyping, or more research has to be done to understand the impact better.

For space limitations, constraints and cross-viewpoint correspondence rules relevant to this viewpoint were omitted in this article.

B. Stakeholder concerns versus decision forces

In the context of ISO/IEC/IEEE 42010, the term *concern* was chosen to include any interest that stakeholders consider fundamental to the architecture of the system (including the process of creating the architecture): "Concerns arise throughout the life cycle from system needs and requirements, from design choices and from implementation and operating considerations." [4]. The standard introduces stakeholders' concerns as a means to drive the selection of architecture viewpoints, i.e. different stakeholders for the architecture description have different needs in terms of different kinds of information. Therefore, concerns result in selecting appropriate representations of the architecture. Forces, in contrast, do not drive representational choices but architecture decisions. The concept of a force is related to the concept of a concern, in that all forces are classified by concerns (see Figure 3). If a force could not be classified by at least one concern, this means that it would not represent any interest of the relevant stakeholders.

IV. Three case studies

To validate the usage of the decision forces viewpoint in software projects, we conducted a multiple-case study with senior students working on non-academic software projects. A case study was preferable over surveys or experiments, because the phenomenon (i.e. the influence of the forces view documentation) had to be studied over a long period of time, thus limiting the possibility for strict control of independent variables [10]. Additionally, a multiple-case design is regarded as more robust than single-case studies, because conclusions from one case can be compared to other cases [11], which increases external validity.

A. Study goal and research questions

Following Robson's classification scheme [12], this multiple-case study is exploratory in nature. The goal is to explore the support provided by the decision forces viewpoint to software architecture activities and the coverage of decision-related concerns in software projects. In particular, the study aims at answering the following two research questions:

- **RQ1**: How does the forces viewpoint support the decision making process?
- **RQ2**: Which of the decision-related concerns mentioned in Table I does the forces viewpoint support?

B. Study design and execution

1) Case descriptions: The study was conducted in the context of two lecturing modules in the software engineering study program at the Fontys University of Applied Sciences in Venlo, the Netherlands. In total, we observed three student groups working on different projects. Two of the projects were conducted as part of a lecturing module, in which student groups work on tasks for external, industrial customers[2]. The third project was done as part of a lecturing module on the Java enterprise edition (JEE). In this module, the students were free to make up their own software project, as long as it involved at least one technology from the JEE specification set. In all cases, the students worked on their own responsibility without lecturers intervening in their decision making process. The decision documentation was no integral part of the modules and was not graded. One of the authors was involved in the third case as a lecturer, while none of the authors was involved in the former two cases. All projects were observed over a period of seven weeks. In the following, the three projects are briefly described:

- **PrjA**: This project is a further development of a legacy documentation system used to generate different types of documents based on templates and dynamically allocated data. The software project was commissioned by a medium-sized German software company. A prominent user of the system is the Bavarian Department of Justice. The primary task of the project group was an architectural re-design to a service oriented architecture, including the migration of the existing functionality to services and the choice and usage of an appropriate enterprise service bus technology.
- **PrjB**: This software was ordered by a Dutch company that acts as a broker between restaurant owners and cooking personnel, specialized on catering, cooking workshops, and interim executive chefs. The student group had to develop a web application for personnel services in the gastronomy business, allowing freelancing cooks to register and apply for jobs. Job offers can be posted by restaurant owners, for instance. The software had to be developed from scratch.
- **PrjC**: The third project was conducted as part of a lecturing module on JEE. The students in this group started an open source project called */notes* (pronounced Slashnotes) for managing, sharing and distributing notes. The software offers three different clients that can be used to access notes: a web application based on JQuery, a Java desktop application (using Swing), and a mobile client for Google's Android operating system. All architecture decisions had to be made by the students. A short video showing the main features of the application can be found on YouTube (http://youtu.be/wW1Lgq2gZvg).

2) Subjects: The subjects of the study were students from the last year of a four-year software engineering program of study. All of the students had already gained some industrial experience from a five-month internship; some of them had additionally pursued part-time jobs in the software engineering industry. During the course of their study, the students had followed different courses on programming, object-oriented analysis and design, and software engineering process models (e.g. RUP, Scrum, Iterative waterfall). To gather their experience regarding programming, design, and software architecture; as well as the time they had already spent in the industry, we

[2]The customers have asked us to stay anonymous.

Table III
PREVIOUS EXPERIENCE OF THE SUBJECTS

	PrjA	PrjB	PrjC
No. stud.	6	5	4
Prog. exp.	75,33 (48,89)	49,2 (13,26)	59,5 (21,56)
Des. exp.	50,33 (28,63)	32,2 (3,03)	33,5 (8,54)
Arch. exp.	38,67 (4,84)	28,6 (9,48)	11,25 (7,97)
Ind. exp.	25,17 (36,21)	7 (2,83)	7,25 (6,18)

asked all participants to fill in a web-based questionnaire prior to the study. Table III shows the number of students in each group (No. stud.), as well as the average number of months of experience that the students had as programmers (Prog. exp.), as software designers (Des. exp.), with software architecture (Arch. exp); and as software engineers in the industry, or as payed freelancers. The numbers in parentheses show the standard deviations. With the exception of one outlier in PrjA regarding programming, design, and industrial experience, the students' previous experiences was comparable between the groups, which renders them equivalent data sources. The fact that the students were in the last semester before the graduation project, and had some first experiences in the IT industry, makes them suitable subjects for the population of inexperienced software engineers at the beginning of their professional careers.

Carver et al. provide a checklist for conducting empirical studies with students [13]. This checklist was used to ensure that the study had a pedagogical value for the participating students and that the results are generalizable to a larger population (in this case the population of inexperienced software engineers). In the following, we list all items of this checklist together with a brief explanation on how the checklist item was considered:

1) **Ensure adequate integration of the study into the course topics** – In both lecturing modules, the students had to make architecture decisions autonomously. The decision forces view supports the decision making process and provides decision-force traceability. Thus, it integrated well into the course topics.
2) **Integrate the study timeline with the course schedule** – The timeline for the study was explicitly planned according to the start of the lecturing modules.
3) **Reuse artifacts and tools where appropriate** – The students used a spreadsheet application for creating the decision forces view. No special tool was introduced for the purpose of decision documentation.
4) **Write up a protocol and have it reviewed** – A study protocol was written before the study and reviewed by the authors in multiple iterations.
5) **Obtain subjects' permission for their participation in the study** – Prior to the two courses, the students were asked if they wanted to participate in the study. They were ensured that no personal data would be made available in the study report. All students expressed their interest in the study. They were also given the opportunity to withdraw from the study by sending an email to the course lecturers.
6) **Set subject expectations** – The students were informed about the effort, we estimated for the decision documentation. Apart from that, we told them that we would give them feedback about how to improve their individual architecting processes after the study.
7) **Document information about the experimental context in detail** – The context of the study is documented in this article.
8) **Implement policies for controlling/monitoring the experimental variables** – The relevant previous experience of the subjects, as well as the descriptions of the projects they were involved in, are reported in this paper. The data collection methods and data sources used to monitor these variables are described in Section IV-B3.
9) **Plan follow-up activities** – At the end of the semester, the students were informed about the study results. Each project group also received individual feedback on their architecting process.
10) **Build or update a lab package** All collected data was stored in a digital study database (as proposed in [11]). The database was used as a basis for the analysis.

3) Data collection: The data collected in this case study is qualitative in nature. We applied triangulation of data-sources, which is a well-accepted method to increase the precision of studies that mainly collect qualitative data [10], [11], [14]. The different data sources that were triangulated, correspond to different data collection methods, which are as follows:

Work artifacts: In the two lecturing modules, from which we recruited our project groups, the students were obliged to store all project related files in Subversion repositories. The researchers were given read access to these repositories, enabling them to track the progress and the iterative refinement of the architectural design.

Focus groups: At the end of the seven weeks, we conducted focus groups with each of the projects. Focus groups are group interviews with a small number of participants, in which a moderator asks questions to concentrate the discussion on a predefined topic. In contrast to individual interviews, focus groups allow group members to build up on each others' answers leading to more profound information [15]. All focus groups were audio recorded and transcribed.

Participant observation: During the seven weeks, the three groups were regularly (at least weekly) visited during their working sessions. The researchers took written notes (i.e. field notes) about their observations, which were afterwards scanned and stored in the study database.

4) Pilot study: To fine-tune the design of the study, in particular the data collection procedures and the research questions, we performed a pilot study. In this pilot study, we used the decision framework, and the forces viewpoint in

particular, to document the architecture decisions of a system for online banking and accounting. One of the authors was involved in the project as a developer. Figure 1 shows an excerpt from the forces view created in this pilot.

The pilot study was particularly helpful for understanding how the forces viewpoint can support the decision making process. In addition, the results were used to develop the question guide, which was employed during the focus groups to ensure that no important topics of interest were forgotten.

C. Analysis procedure and results

As the data in our study database was qualitative to a large extend, we chose to apply a grounded theory approach [16] to analyze the data. While being used mainly in social sciences, grounded theory has recently also gained more attention in software engineering related research (see for instance [17], [18]).

1) Analysis procedure: Grounded theory is inherently explorative in nature, as it promotes the analysis of data without predetermined ideas about potential findings. Concepts emerge slowly by constantly comparing indicators found in the data to previously identified indicators. That way, an idea about a finding (usually referred to as a theory) is either supported by additional evidence, or it has to be rejected, if no additional indicators can be found to carry it. In the following, the steps we followed during the data analysis are briefly described. Note that steps two to four are performed iteratively.

1) **Convert data to PDF**: The gathered data was exclusively stored digitally. As a preparation for the data analysis, we converted all files in the study database to the PDF format to allow for a uniform coding procedure.
2) **Coding**: All PDFs were intensively studied. Indicators for concepts related to decision views (in particular the forces view) were coded (i.e. labelled) as brief statements using PDF annotations. Please refer to Adolph et al. [17] for an extensive explanation of the terms *indicator*, *code*, *concept*, and *category*, which are central concepts in grounded theory.
3) **Identify concepts**: During the coding procedure, concepts emerge, which represent candidate patterns of behavior, suggested by a set of indicators. The concepts were registered and related to the codes supporting it. The result after some iterations of analysis, was a set of concepts describing how the three student groups used and perceived the forces viewpoint in their projects.
4) **Classify concepts into categories**: Finally, in the last step of the analysis, the concepts from the three groups were compared to identify common categories of concepts. A category is a concept on a higher level of abstraction. As stated above, findings that were concordantly made in more than one project group are more reliable.

2) Analysis and interpretation: Table IV summarizes the results of the qualitative analysis. The table maps the categories, identified in step 4 of the analysis procedure, to the project groups, in which they were observed. Additionally, the table shows decision-related concerns that are related to some of the categories, as well as research questions (column *Res. Qu.*), to which the categories contribute. In the following, the results are interpreted in the context of the two research questions. The interpretation focuses on categories that were recognized in at least two of the projects; only regarding suggestions for improvement, we discuss categories assigned to single groups only.

RQ1: How does the forces viewpoint support the decision making process? As Table IV shows, the data collected from all three groups indicated that the forces views caused the students to take the decision making process more seriously than they would have done otherwise (**Cat1**). The fact that decisions and forces had to be documented explicitly caused the students to think more concretely about available decision alternatives (**Cat5**), and the forces that influence the choice between these alternatives. The students noticed that the view prevented them from making decisions ad-hoc (**Cat3**, **Cat19**). A comment in a focus group was "If you don't have the view, then you might also see alternatives, but if I have experience in a solution then I will choose this one. But with the (forces) view, you are forced to think about which one is really better." It is notable that all groups mentioned that the forces views triggered them to consider quality attribute requirements in the first place (**Cat2**). They had not thought of this in projects before (during their studies or in side jobs). Among all collected work artifacts, the forces views were the only documents in which quality attributes were mentioned. Considering quality attributes in architectural design, however, is an important best-practice that should be adopted by inexperienced software engineers.

In general, the forces viewpoint was very well received by the students. They found it especially helpful to maintain an overview over decisions made and the factors that influence the decisions (**Cat18**). The majority of members in all groups explicitly stated that they will reuse the forces viewpoint in future projects (**Cat4**)[3]. They acknowledged that it is a good way of documenting architecture decisions (**Cat6**). This finding is particularly important, because our experience from multiple studies with students shows that they cannot be convinced to document their decisions using decision templates (e.g. from [2]). They usually perceive decision documentation as a tedious task that does not have an immediate benefit. The forces viewpoint, in contrast, is a documentation approach that they quickly accepted; presumably because of its relative lightweightiness and its immediate support for the decision making process.

Although the students were predominantly positive about the forces viewpoint, they also made suggestions for improvement. ProjectC was concerned about the fact that the forces viewpoint does not provide means to specify different weights for forces (**Cat14**). In their project, some forces were clearly

[3]At the time this paper was written, the students were working on their final bachelor projects in external companies. We repeatedly received questions and suggestions about the forces viewpoint, which indicates that at least some students indeed keep using decision views.

Table IV
RESULT OF THE QUALITATIVE ANALYSIS

Code	Category	PrjA	PrjB	PrjC	Concerns	Res. Qu.
Cat1	Required students to think more carefully about decisions.	X	X	X		RQ1
Cat2	Triggered students to consider quality attribute requirements.	X	X	X		RQ1
Cat3	Prevents ad-hoc decisions.	X	X	X		RQ1
Cat4	Forces viewpoint will be used in other projects.	X	X	X		RQ1
Cat5	Triggered students to identify more alternatives.	X	X			RQ1
Cat6	Good way to document decisions.		X	X		RQ1
Cat7	Creating the forces view took a lot of time.	X				RQ1
Cat8	Prevents inefficient discussions about decisions.	X				RQ1
Cat9	Created with reasonable effort.	X				RQ1
Cat10	Saved time in the end.		X			RQ1
Cat11	Support for rational decisions.			X		RQ1
Cat12	Forces view complements relationship view.			X		RQ1
Cat13	Useful for architects, designers, programmers, and new project members.			X		RQ1
Cat14	Support for weighing forces is missing.			X		RQ1
Cat15	Identifying all forces is a matter of experience.			X		RQ1
Cat16	Forces view and relationship view are simultaneously refined.			X		RQ1
Cat17	Proper tool support needed.			X		RQ1
Cat18	Maintain overview over architectural decisions, concerns, and forces.	X	X	X	C4,C5,C6	RQ1,RQ2
Cat19	Helpful to systematically compare decision alternatives in the context of forces.	X	X	X	C5,C6	RQ1,RQ2
Cat20	Help for estimating requirements coverage.	X		X	C6	RQ1,RQ2
Cat21	Support for systematic trade-offs between forces.			X	C7	RQ1,RQ2
Cat22	Supports sharing architecture rationale.	X	X	X	C3, C23	RQ2

more important than other forces causing them to select an architecture decision alternative that had a lower rating (i.e. sums of pluses and minuses) than the other alternatives. Although we had considered this aspect during the design of the forces viewpoint, we chose not to include it in the viewpoint specification to keep it simple. Systematically weighing forces would have introduced additional complexity, which could have deterred students from using the view properly. However, the forces viewpoint can easily be customized by stakeholders in order to introduce such weights in their projects. Apart from this, it became evident that identifying all relevant forces is a matter of experience (**Cat15**). Therefore, especially for domain-specific forces, it can be helpful to collect typical forces from different projects that can be used as a checklist to ensure that no important forces are forgotten. Tool support would also be appreciated, especially to ensure consistency and to save work when creating the forces view in addition to other views from the framework (**Cat17**).

RQ2: Which decision-related concerns does the forces viewpoint support? To find out for which decision-related concerns the students used the forces views, we analyzed the concepts and categories and compared them to the list of concerns in Table I. The results are shown in Table IV (categories 18 to 22). Because the categories are conceptually more abstract than single concerns, sometimes multiple concerns are mapped to a single category. Note that the students were not knowledgable about the concerns we had assigned to the forces viewpoint in the specification. This would have introduced a threat to the validity of our findings.

The concepts classified under category **Cat18** have shown that all three groups used the forces views to maintain an overview over architectural decisions, concerns, and forces. The students described that one column in the forces view (see Figure 1) shows which concerns (**Cat18**, concern C4), and which forces (**Cat18**, concern C5) are related to a decision. They also understood that a row in the view shows decisions influenced by a specific force (**Cat18**, concern C6). This information was actively used by the students to make the choice between multiple alternatives more systematic (**Cat19**, concerns C5, C6).

All three groups saw value in the forces viewpoint with respect to sharing architecture rationale (**Cat22**, concern C3). In particular, they mentioned that usually individual members of the groups were more knowledgeable about specific architectural decision alternatives and their relation to forces than others. The forces views helped them to spread this knowledge better among the group members. Using their own words, the student groups stated that studying the forces view helped everybody to understand the why behind architecture decisions, including the decisions primarily made by others. Category **Cat22** was also assigned to concern C23, because the students saw the potential of the forces views to facilitate the reusability of decisions in other projects: by providing the rationale in terms of decisions addressing specific forces, the decisions can be reused in cases where similar rationale would make sense.

Two groups used the forces views to estimate the coverage of some important requirements (**Cat20**). During the analysis of the work artifacts, we could see that all groups had used requirements as forces; only two of the groups, however, had also actively used the forces view to check in how far the decisions made were suitable to actually satisfy the requirements. They understood that a row in the view shows all decisions that need to be regarded when estimating the

coverage of a particular requirement (i.e. a force in the forces view). For the same reasons, **Cat20** confirms concern C6, which is about identifying all decisions that were influenced by a particular force.

Concern C7 (Which forces have conflicting influences on a decision?) was only explicitly approved by one project. Conflicting influences have to be regarded when making trade-offs (**Cat21**). In forces views, conflicting impacts are indicated by a decision that has positive rating for one force and negative ratings for another force. Although this situation was observed in the forces views of all three groups, only one of the groups explicitly acknowledged the usefulness of forces views for making trade-offs. We conjecture that the other groups did not mention trade-offs, because they had not explicitly discussed such situations. Only in PrjC, we observed that the group actively and fully-aware discussed conflicting impacts and ways to compensate resulting issues. This corresponds to the team's earlier discussed statement that they were missing weights for forces (**Cat14**). Particularly when making trade-offs, different weights of forces should be considered.

D. Threats to validity

In the following, we present potential threats to the validity of our findings. In particular, we cover typical validity threats in software engineering studies, as identified in [11] and [19].

1) Construct Validity: Construct validity is concerned with the operational measures taken to analyze the phenomenon under study. In this case, we used multiple sources of evidence (i.e. work artifacts, field observation, and focus groups) to study the use of the forces viewpoint in software projects. Additionally, the use of a grounded theory approach ensures that conclusions are rooted in the collected data and that no important concepts are forgotten.

2) Internal Validity: Internal validity mainly has to be considered in explanatory case studies [11], in which a cause-effect relationship is going to be established. In exploratory case studies, internal validity basically concerns making inferences. In this case, we tried to address this potential threat by involving different sources of data, including direct participant observation and analysis of work artifacts. Logical deductions are generally based on multiple sources of evidence and aligned among at least two of the projects under study (we did not make deductions from data coming from only one project).

3) External Validity: External validity concerns the generalizability of the study's findings to a larger population. Because statistically representative samples can typically not be achieved in cases studies, the emphasis is usually put on analytical generalization, thus an explanation why the findings are representative for other cases with common characteristics [10]. Yin points out that external validity can be improved by using replicated study-designs [11]. In this study report, we present results that are based on findings made in three different cases using identical study designs. This reduces the influence of the concrete cases and of the individual students in the different project groups. Therefore, we assume that our findings are relevant at least for the population of inexperienced software engineers at the beginning of their professional careers. Although we did not find any indicators raising legitimate doubts about the usefulness of the decision forces viewpoint for experienced software architects as well, additional industrial studies must be conducted to generalize the study results to this larger population.

4) Reliability: The reliability of a study is concerned with the minimization of errors and biases that stem from the researchers who conducted the study. In this case, the moderator of the focus groups could have influenced the students towards giving specific answers. This threat was mitigated by asking open questions like "How did the decision forces view influence your decision making process?". As follow-up questions, the moderator asked the students to explain their answers, or to go more into detail. To mitigate the risk of suggestive questions and to make sure that all important topics would be covered, we prepared a question guide [20] in advance, which was used by the moderator during the focus groups.

An additional potential threat to reliability could result from students not staying true to the facts during the focus groups. To mitigate this risk, we used data-source triangulation [21], which allowed us to verify concepts using different types of data. Additionally, as stated above, we prioritize results that were concordantly found in at least two of the three case studies.

V. Related work

The work presented in this paper is related to architecture decision documentation in general, and architecture decision views in particular. In our recent publication, we extensively discussed related work in these two fields [3]. Therefore, in the remainder of this section, we focus on related work with respect to traceability between requirements (problems) and design (solutions).

The decision forces viewpoint acknowledges the importance of relating architecture decisions to the forces driving those decisions. As such, the forces viewpoint is connected to the research area of relating architecture and rationale. In their recent book, Avgeriou et al. compiled 15 articles that relate architecture and requirements [22], taking among others traceability between architecture design, decision rational and requirements into account. Tang et al. in the same publication, provide a traceability metamodel for bridging the gap between elements from the problem space (stakeholders, requirements, and issues) and elements from the solution space (architectural design, structure, components) using architecture decisions and rationale as intermediaries. Other authors had proposed to use reference models to support different types of requirements traceability before (e.g. [23], [24]).

A slightly different approach to software architecture–requirements traceability has recently been introduced by Malavolta et al. [25]. Originating from the model-driven architecture field, they suggest to use weaving models to relate requirements models, architecture decision models, and

different types of architecture descriptions. In contrast to using one shared metamodel, weaving models are non-invasive and provide greater flexibility.

Tang et al. provide an architecture model for design traceability and reasoning [26]. The model connects architecture description elements (as defined in IEEE Std 1471-2000 [7]) to architecture decisions and architecture rationale, as first class entities. These authors also implicitly acknowledge the existence of decision forces, by introducing a concept they call *motivational reason*. A motivational reason can be among others a requirement, a goal, an assumption, or a constraint.

Despite this existing work on architecture rationale-design traceability, to the best of our knowledge, no approach exists that systematically integrates this traceability in a software architecture description following the conventions of ISO/IEC/IEEE 42010. Additionally, and more importantly, very few authors have recognized the importance of treating the full scope of decision forces extending across the context of the system and the environment in which it is developed, as first-class entities in an architecture description. We argue that the concept of decision forces, as introduced here, is a valuable contribution to the field.

VI. Conclusions and future work

In this paper, we introduced the decision forces viewpoint as an extension to our framework for documenting architecture decisions. The viewpoint was validated in a multiple-case study, which has shown that the forces viewpoint is very well received, while satisfying its related concerns. Additionally, the forces viewpoint has demonstrated its ability to support inexperienced software engineers during the decision making process, by providing a structure that triggers them to consider multiple architectural decision alternatives and systematically compare them in the context of all important forces.

We are currently observing the use of the forces viewpoint and other decision viewpoints from our framework in an industrial study, in which we analyze the suitability of decision views for problem and design space documentation. Apart from that, we have used it as part of a decision-based architecture evaluation method, which we are currently developing.

Finally, as suggested by many users of our decision viewpoints, we continue the development of a tool suite, which efficiently supports architects in documenting views corresponding to our viewpoints.

Acknowledgements

We would like to thank all participating students from the software factories and the Java enterprise edition course 2011/2012. Two of the cases reported on in this paper are part of a larger study designed and conducted together with Antony Tang.

We would also like to thank Veli-Pekka Eloranta and Kai Koskimies, with whom we initially discussed the concept of decision forces in the context of decision-based architecture evaluation.

References

[1] P. Kruchten, "An ontology of architectural design decisions in software intensive systems," in *Proceedings of the 2nd Groningen Workshop on Software Variability*, 2004, pp. 54–61.

[2] J. Tyree and A. Akerman, "Architecture Decisions: Demystifying Architecture," *IEEE Software*, vol. 22, no. 2, pp. 19–27, 2005.

[3] U. van Heesch, P. Avgeriou, and R. Hilliard, "A documentation framework for architecture decisions," *Journal of Systems and Software*, vol. 85, no. 4, pp. 795 – 820, 2012.

[4] *ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description*, ISO, December 2011.

[5] D. Perry and A. Wolf, "Foundations for the study of software architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, 1992.

[6] P. Kruchten, "The 4+ 1 View Model of Architecture," *IEEE Software*, vol. 12, no. 6, pp. 42–50, 1995.

[7] *IEEE Std 1471–2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, IEEE, October 2000.

[8] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-oriented software architecture: a system of patterns*. John Wiley & Sons, Inc. New York, NY, USA, 1996.

[9] G. Mustapic, A. Wall, C. Norstrom, I. Crnkovic, K. Sandstrom, J. Froberg, and J. Andersson, "Real world influences on software architecture-interviews with industrial system experts," in *Fourth Working IEEE/IFIP Conference on Software Architecture, 2004. WICSA 2004*. IEEE, 2004, pp. 101–111.

[10] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[11] R. K. Yin, *Case Study Research: Design and Methods, Applied Social Research Methods Series, Vol 5*, 5th ed. Sage Inc., 2009.

[12] C. Robson, *Real world research*. Wiley, 2011.

[13] J. Carver, L. Jaccheri, S. Morasca, and F. Shull, "A checklist for integrating student empirical studies with research and teaching goals," *Empirical Software Engineering*, vol. 15, no. 1, pp. 35–59, 2010.

[14] R. Stake, *The art of case study research*. Sage Publications, Inc, 1995.

[15] J. Kontio, J. Bragge, and L. Lehtola, "The focus group method as an empirical tool in software engineering," *Guide to advanced empirical software engineering*, pp. 93–116, 2008.

[16] B. Glaser and A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Publishing Company, 1967.

[17] S. Adolph, W. Hall, and P. Kruchten, "Using grounded theory to study the experience of software development," *Empirical Software Engineering*, vol. 16, no. 4, pp. 487–513, 2011.

[18] C. Urquhart, H. Lehmann, and M. Myers, "Putting the theory back into grounded theory: guidelines for grounded theory studies in information systems," *Information systems journal*, vol. 20, no. 4, pp. 357–381, 2010.

[19] C. Wohlin, M. Hoest, P. Runeson, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering: an introduction*. Kluwer Academic Pub, 2000.

[20] N. Mack, C. Woodsong, K. MacQueen, G. Guest, and E. Namey, *Qualitative research methods: A data collector's field guide*. FLI, 2005.

[21] T. Lethbridge, S. Sim, and J. Singer, "Studying Software Engineers: Data Collection Techniques for Software Field Studies," *Empirical Software Engineering*, vol. 10, no. 3, pp. 311–341, 2005.

[22] P. Avgeriou, J. Grundy, J. Hall, P. Lago, and I. Mistrik, *Relating Software Requirements and Architectures*. Springer Publishing Company, Incorporated, 2011.

[23] O. Gotel and C. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of the First International Conference on Requirements Engineering*. IEEE, 1994, pp. 94–101.

[24] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE Transactions on Software Engineering*, vol. 27, no. 1, pp. 58–93, 2001.

[25] I. Malavolta, H. Muccini, and V. Smrithi Rekha, "Supporting architectural design decisions evolution through model driven engineering," *Software Engineering for Resilient Systems*, pp. 63–77, 2011.

[26] A. Tang, Y. Jin, and J. Han, "A rationale-based architecture model for design traceability and reasoning," *Journal of Systems and Software*, vol. 80, no. 6, pp. 918–934, 2007.